A Biologically Inspired Model with Fusion and Clonation of Membranes

Giorgio Delzanno[1] and Laurent Van Begin[2*]

[1] Università di Genova, Italy, giorgio@disi.unige.it
[2] Université Libre de Bruxelles, Belgium, lvbegin@ulb.ac.be

Abstract. P-systems represent an important class of biologically inspired computational models. In this paper, we study computational properties of a variation of P-systems with rules that model in an abstract way fusion and clonation of membranes. We focus our attention on extended P-systems with an interleaving semantics and symbol objects and we investigate decision problems like reachability of a configuration, boundedness (finiteness of the state space), and coverability (verification of safety properties). In particular we use the theory of well-structured transition systems to prove that both the coverability and the boundedness problems are decidable for PB systems with fusion and clonation. Our results represent a preliminary step towards the development of automated verification procedures for concurrent systems with biologically inspired operations like fusion and clonation.

1 Introduction

In recent years several efforts have been spent to define unconventional computing models inspired by biological systems. One interesting family of this kind of models is that of P-systems [15]. P-systems are a basic model of the living cell defined by a set of hierarchically organized membranes and by rules that dynamically distribute elementary objects in the component membranes. Several variations of the basic model have been proposed in the literature, e.g., with active membranes [16], with string objects [15], with dissolution [7], division [16], and gemmation rules [4]. The PB-systems of Bernardini and Manca [3] represent one of the variants of the basic model in which rules can operate on the boundary of a membrane. A *boundary rule* can be used to move multisets of objects across a membrane. In biological modelling, PB-systems can be used to express complex interactions among biological membranes [10].

In this paper we take PB-systems as a starting point for studying computational properties of an extension of P-systems with two biologically inspired operations, namely *fusion* and *clonation* of membranes. Membrane fusion is defined in other artificial models of the living cell like the bio-ambients of Regev et al. [18], a model based on process algebra. A restricted form of division, namely

* Research fellow supported by the Belgian National Science Foundation (FNRS).

replication, is also provided in biologically inspired models based on process algebra. The replication operation $!P$ is defined (via the congruence $!P \equiv P|!P$) in such a way that an arbitrary number of copies of process P can be generated during the execution of a system. Since the process $!P$ cannot interact with the environment, replication can be viewed as an operation that dynamically injects new processes in their *initial state*. Differently from replication, we consider here a clonation operation in which a membrane can generate a copy of its *current state* (that may include other membranes). This unconventional form of dynamic process generation is inspired by biological processes. Indeed, it can be viewed as an abstract version of the *division* operation introduced in the context of P-systems in [17].

In this paper we focus our attention on decision problems related to basic qualitative properties of our extended notion of PB systems (PBFC systems, for short). Specifically, we investigate the decidability of properties like *reachability* of a given configuration, *boundedness* (finiteness of the state space), and *coverability* (reachability of an infinite set of configurations sharing a given pattern). We study all these properties using an interleaving operational semantics (i.e., no maximal parallelism) with no priorities. Furthermore, we consider membranes containing multisets (i.e., unordered collections) of objects. These limitations allows us to explore the expressiveness of our biologically inspired model independently from specific execution strategies, and additional ordering on elementary objects. Similar decision problems for qualitative analysis of subclasses of P-systems have been studied, e.g., in [14, 6, 7].

Our technical results are as follows. We first show that reachability for PBFC is undecidable. This result follows from a *weak* encoding of multi-counter machines in PBFC. The encoding is weak in the following sense: some execution paths of the PBFC that simulates the counter machine may take a *wrong* turn into a path that does not correspond to a simulation of the original model. We can use however information inserted in the target configuration of a reachability problem to restrict the simulation to *good* paths only. The encoding exploits the property that the set of reachable configurations of a PBFC model may contain configurations with unbounded width (due to the presence of clonation) and with multisets of objects of unbounded size (e.g., due to the presence of internal and boundary rules). This property however is not sufficient to obtain a Turing equivalent model. Indeed, we show that boundedness and coverability are both decidable for PBFC systems. These results are based on the theory of *well-structured transition systems* developed in [1, 9] for the analysis of infinite-state (concurrent) systems. Such a theory finds here a novel application to unconventional computing models.

The introduction of maximal parallelism and/or priorities would lead to a Turing complete model as in the case of PB-systems. The analysis of a model with interleaving semantics represents however a promising preliminary step towards the development of automated verification procedures for concurrent models similar to ours with unconventional (biologically inspired) primitives like fusion and clonation.

Related work To our current knowledge, these are the first (un)decidability results obtained for qualitative analysis of extensions of PB systems with both fusion and clonation rules, interleaving semantics and symbol objects. Decidability results for basic PB systems have been obtained in [6, 7]. Specifically, in [6] Dal Zilio and Formenti proved that reachability, boundedness and coverability are decidable for PB systems with symbol objects by using a reduction to Petri nets. In [7] we proved that reachability is still decidable for an extension of PB systems with creation of new membranes with fixed content (e.g., an empty membrane) or with membrane dissolution, but not both. Interestingly, boundedness and coverability turn out to be undecidable with creation rules. We consider here operations of different nature (e.g., clonation cannot force the newly created membrane to be empty and does not increase the depth of configurations). The resulting extension of PB systems satisfies indeed different properties (undecidability of reachability, decidability of coverability and boundedness). The universality problem of different form of division in P-systems with active membranes has been studied in [17, 2]. Differently from [17, 2], we consider here a more abstract notion of division, we called clonation, and different decision problems like coverability more related to verification of qualitative problems. Similar verification problems have been investigated for other variations of P-systems (e.g., signaling and catalytic P-systems) in [14, 12].

2 Preliminaries

In this section we recall the main definitions of well-structured transition systems [1, 9], and PB-systems [10]. We first need some preliminary notions. Let $\mathbb{N}$ be the set of positive integers. Consider a finite alphabet Γ of symbols. A multiset over Γ is a mapping $u : \Gamma \mapsto \mathbb{N}$. For any $a \in \Gamma$, the value $u(a)$ denotes the multiplicity of a in u (the number of occurrences of symbol a in u). We often use a multiset as a string $a_1 \cdot \ldots \cdot a_n$ of symbols, i.e., $a_i \in \Gamma$. Furthermore, we use ϵ to denote the empty multiset, i.e., such that $\epsilon(a) = 0$ for any $a \in \Gamma$. As an example, for $\Gamma = \{a, b, c, d\}$, the string $abcc$ represents the multiset u such that $u(a) = u(b) = 1$, $u(c) = 2$, and $u(d) = 0$. We use $\Gamma^{\otimes}$ to denote the set of all possible multisets over the alphabet Γ. Given two multisets u, v over Γ, we write $u \preceq v$ if $u(a) \leq v(a)$ for all $a \in \Gamma$. We use $u \prec v$ to denote that $u \preceq v$ and $v \not\preceq u$. Furthermore, we use $\oplus$ and $\ominus$ to denote multiset union and difference, respectively. Specifically, for any $a \in \Gamma$ we have that $(u \oplus v)(a) = u(a) + v(a)$, and $(u \ominus v)(a) = max(0, u(a) - v(a))$ where $max(a, b)$ returns the largest number between a and b.

Well-structured transition systems. A *transition system* is a tuple $G = (S, T)$ where S is a (possibly infinite) set of configurations, $T \subseteq S \times S$ is a transition relation between configurations. We use $\gamma \to \gamma'$ to denote $(\gamma, \gamma') \in T$. A quasi-ordering $(S, \sqsubseteq)$ is a *well-quasi ordering* (wqo for short) if for any infinite sequence $s_1 s_2 \ldots s_i \ldots$ there exist indexes $i < j$ such that $s_i \sqsubseteq s_j$. A transition system $G = (S, T)$ is a *well-structured transition system* (wsts for short) with respect to a quasi-order $\sqsubseteq \subseteq S \times S$ iff: (i) $\sqsubseteq$ is a well-quasi ordering; and (ii) for any configurations $\gamma_1, \gamma_1', \gamma_2$ such that $\gamma_1 \sqsubseteq \gamma_1'$ and $\gamma_1 \rightarrow \gamma_2$, there exists γ_2' such that $\gamma_1' \rightarrow \gamma_2'$ and $\gamma_2 \sqsubseteq \gamma_2'$, i.e., G is *monotonic*. A wsts is said *strictly monotonic* when $\gamma_1 \sqsubset \gamma_1'$, i.e., $\gamma_1 \sqsubseteq \gamma_1'$ and $\gamma_1' \not\sqsubseteq \gamma_1$, implies that $\gamma_2 \sqsubset \gamma_2'$.

P-systems with Boundary Rules. A PB system [3] with symbol objects is a tuple $\Pi = (\Gamma, N, M, R, \mu_0)$, where:

- Γ is a finite alphabet of symbols.
- N is a finite set of membrane names/types.
- M is a finite tree representing the *membrane structure*. Each node n of M corresponds to a membrane and is labelled with a membrane name/type $\mathsf{type}(n) \in N$. We use $\mathsf{nodes}(M)$ to denote the set of nodes of M.
- R is a finite set of rules.
- $\mu_0 : \mathsf{nodes}(M) \mapsto \Gamma^{\otimes}$ is the *initial configuration*, i.e., a mapping from membranes to multisets of objects from Γ.

Rules can be of the following two forms[1]:

$$\begin{array}{lll} (1) & \text{Internal:} & [_i\ u \rightarrow [_i\ v \\ (2) & \text{Boundary:} & u\ [_i\ v \rightarrow u'\ [_i\ v' \end{array}$$

where $i \in N$, and $u, u', v, v' \in \Gamma^{\otimes}$ and we assume in boundary rules that at least one between u and u' is not empty.

The semantics of PB-systems is given in terms of transition systems. The set of configurations of a PB system Π is the set of distributions of objects in Γ in the membranes in M, i.e., a configuration μ is mapping from $Nodes(M)$ to $\Gamma^{\otimes}$.

The transition relation is defined as follows. A rule of the form (1) is enabled at μ, if there exists a membrane $n \in \mathsf{nodes}(M)$ with $\mathsf{type}(n) = i$ and $u \preceq \mu(n)$. Its application leads to a new configurations μ' such that

$$\mu'(n) = (\mu(n) \ominus u) \oplus v$$

and $\mu'(n') = \mu(n')$ for any other node $n' \in \mathsf{nodes}(M)$ such that $n \neq n'$.

Suppose now that a membrane $m \in \mathsf{nodes}(M)$ with $\mathsf{type}(m) = j$ contains as immediate successor in M a node n with $\mathsf{type}(n) = i$. A rule of the form (2) is enabled at μ, if $u \preceq \mu(m)$ and $v \preceq \mu(n)$. Its application leads to a new configuration μ' such that

$$\begin{array}{l} \mu'(m) = (\mu(m) \ominus u) \oplus u' \\ \mu'(n) = (\mu(n) \ominus v) \oplus v' \end{array}$$

and $\mu'(m') = \mu(m')$ for any node $m' \in \mathsf{nodes}(M)$ such that $m' \neq m, n$. We have a transition from μ to μ', i.e., $\mu \rightarrow \mu'$, if μ' can be obtained from μ by applying a rule in R. A computation with initial configuration μ_0 is a sequence of transitions $\mu_0 \rightarrow \mu_1 \rightarrow \ldots \mu_k$. A configuration μ is reachable from μ_0 if there exists a computation $\mu_0 \rightarrow \mu_1 \rightarrow \ldots \mu_k$ with $\mu_k = \mu$.

[1] We consider here a slight generalization of the model in [6] in which we allow any kind of transformation between two membranes.

3 PB systems with Fusion and Clonation

In this paper, we investigate an extension of PB-systems with two new operations namely fusion and clonation of membranes. On one side these two new operations allow to describe basic behaviors of a living cell. On the other side they complicate the formalism in that the membrane structure may dynamically change during the evolution. The state space of an instance of the extended model may be infinite in two dimensions: in the size of the membrane structures generated during the evolution of the system, and in the number of objects produced inside the corresponding membranes.

Formally, a PB system with *fusion* and *clonation* rules (PBFC, for short) provides, in addition to internal and boundary rules, two other kinds of rules of the following form:

$$\begin{array}{ll} (3)\ \text{Fusion}: & [_i\ u\ \ [_j\ v\ \ \to [_k\ w \\ (4)\ \text{Clonation}: & [_i\ u\ \ \ \to [_i\ v\ \ [_i\ w \end{array}$$

where $i, j, k \in N$, and $u, v, w \in \Gamma^{\otimes}$. The rule (3) models the fusion of a membrane m with $\mathsf{type}(m) = i$ containing the multiset of objects u with a membrane m' with $\mathsf{type}(m') = j$ containing the multiset of objects v. Objects in u and v are consumed during this process. The fusion of the two membranes generates a new membrane n with $\mathsf{type}(n) = k$ that contains w and the (remaining) contents of both m and m'. A clonation rule like (4) clones a sub-tree rooted by a membrane n with $\mathsf{type}(n) = i$ containing the multiset of objects u. During the clonation, the objects in u are consumed and replaced by the multiset of objects v in n and by the multiset of objects w in the clone of n. This definition allows to define both perfect clones (i.e., two copies of the same membrane) or to distinguish the clone from the original membrane by using objects w and v, respectively. The latter type of clonation can be used to disable a second application of clonation immediately after the generation of the clone (i.e., avoid clonation rules that are enabled forever and thus applied without control).

To make the semantics formal, we make the membrane structure part of the current configuration, M_0 being the initial tree. Thus, a configuration is now a pair $c = (M, \mu)$, where M is a tree, and $\mu : \mathsf{nodes}(M) \mapsto \Gamma^{\otimes}$ is a mapping from nodes of M to $\Gamma^{\otimes}$. Rules of type (1) and (2) operate on a configuration $c = (M, \mu)$ without changing the tree structure M and changing μ as specified in the semantics of PB systems.

A fusion rule like (3) operates on a configuration $c = (M, \mu)$ as follows. Suppose that m and m' are two descendants of a node p in M such that $\mathsf{type}(m) = i$ and $\mathsf{type}(m') = j$. The rule is enabled if $u \preceq \mu(m)$ and $v \preceq \mu(m')$. Its application leads to a new configurations $c' = (M', \mu')$ such that

- M' is the tree obtained by removing the nodes m and m', adding a new node n with $\mathsf{type}(n) = k$, and by letting all successor nodes of m and m' become successors of n. The parent node of n is p, the parent of the nodes m and m' in the tree M;

– μ' is the mapping defined as

$$\mu'(n) = (\mu(m) \ominus u) \oplus (\mu(m') \ominus v) \oplus w$$

and $\mu'(n') = \mu(n')$ for any other node $n' \in \mathsf{nodes}(M')$ such that $n' \neq n$.

A clonation rule like (4) operates on a configuration $c = (M, \mu)$ as follows. Suppose that M has a node m with a successor n with $\mathsf{type}(n) = i$. The rule is enabled if $u \preceq \mu(n)$. Its application leads to a new configuration $c' = (M', \mu')$ such that

– M' is the tree obtained by adding a new copy of the tree rooted by n as sub-tree of m;
– μ' is the mapping defined as follows. For any node n' in the sub-tree rooted by n, let $\mathsf{Clone}(n')$ be its copy in the new sub-tree. Then, we have that
  - $\mu'(n) = (\mu(n) \ominus u) \oplus v$;
  - $\mu'(\mathsf{Clone}(n)) = (\mu(n) \ominus u) \oplus w$;
  - $\mu'(\mathsf{Clone}(n')) = \mu(n')$ for any node $n' \neq n$ in the sub-tree rooted by n;
  - $\mu'(n') = \mu(n')$ for the other nodes $n' \in \mathsf{nodes}(M')$.

The notions of computation and reachable configuration can naturally be extended to PBFC systems.

Example 1. Consider a PBFC system with $\Gamma = \{a, b, c, d, e, \ f, g, h, u, v, w\}$, $N = \{i\}$. For simplicity, configurations are represented here as terms. Specifically, objects are represented as constants, and a membrane of type i containing $t_1, \ldots, r_n$ as a term of the form $[t_1, \ldots, t_n]$. Hence, $[a\ [b]\ [c]]$ represents the configuration where the root node contains the object a and two membranes, with objects b and c, respectively. Now consider the initial configuration $[a[b[d]][c]]$ and the following set of rules:

$$\begin{array}{lll}
(r_1) & [_i\ d \to [_i\ f\ \ [_i\ g & (\mathit{clonation}) \\
(r_2) & b\ [_i\ g \to e\ [_i\ h & (\mathit{boundary}) \\
(r_3) & [_i\ e \to [_i\ v\ \ [_i\ u & (\mathit{clonation}) \\
(r_4) & [_i\ u[_i\ c \to [_i w & (\mathit{fusion})
\end{array}$$

Then, we have the following computation:

$$\begin{array}{l}
[a\ [b\ [d]]\ [c]] \to_{r_1} [a\ [b\ [f]\ [g]]\ [c]] \to_{r_2} [a\ [e\ [f]\ [h]]\ [c]] \\
\to_{r_3} [a\ [v\ [f]\ [h]]\ [u\ [f]\ [h]]\ [c]] \to_{r_4} [a\ [v\ [f]\ [h]]\ [w\ [f]\ [h]]]
\end{array}$$

Decision Problems In this paper, we focus our attention on decision problems related to the dynamic behavior of PBFC systems. The first problem we consider is the reachability of a configuration.

Definition 1 (Reachability). *Given a PBFC system Π with the initial configuration (M_0, μ_0) and a configuration (M, μ), the* reachability problem *consists in checking if (M, μ) is reachable from (M_0, μ_0).*

The reachability problem is decidable for PB systems with symbol objects [6], and it remains decidable for PB systems with dissolution [7], and for PB systems with dynamic creation of membranes [7]. When adding fusion and clonation, two counter machines may be weakly simulated with PBFC in the sense that some executions of PBFC systems do not simulate an execution of the counter machine. However, in those cases a particular membrane contains objects, allowing to distinguish those executions from the one simulating the counter machine by looking at the content of that membrane in the last configuration of the execution. Hence, the reachability problem is undecidable for PBFC system.

Theorem 1. *The reachability problem is undecidable for PBFC systems.*

Proof. We reduce the reachability problem for two counter machines. Our reduction uses the following types of membrane: cs (for control structure), c_1, c_2, and $trash$. Each configuration has a root node of type cs. Counters are encoded with membranes of type c_1 and c_2. Those membranes and one trash membrane are the children of the root. The trash membrane is used to validate executions. The set of objects contains the control states of the counter machine and their primed version, i.e. the root membrane containing control state l means that the counter machine is in control state l. The primed versions correspond to intermediate states. We also have objects o that are used to encode the value of counters, i.e., a membrane of type c_1 with k objects o represents the value k for the first counter, and objects $active_1$, $active_2$, $clone_1$, $clone_2$, $fusion$ and $fused$. The six last objects are useful for simulation. The initial configuration with control state l_0 and both counters equal to 0 is encoded with a configuration where the root (of type cs) contains l_0, the child of type c_i contains the object $active_i$ and the trash membrane is empty. An increment of counter i from location l_1 to location l_2 is simulated by a rule

$$l_1 \ [_{cs} \ active_i \rightarrow l_2 \ [_{cs} \ active_i \cdot o$$

A decrement of counter i from location l_1 to location l_2 is simulated by a rule

$$l_1 \ [_{cs} \ active_i \cdot o \rightarrow l_2 \ [_{cs} \ active_i$$

Finally, a zero test on counter i and a move from l_1 to l_2 is simulated by four rules. The two first ones clone membrane c_i:

$$l_1 \ [_{cs} \ active_i \rightarrow l_1' \ [_{cs} \ clone_i$$

$$[_{c_i} clone_i \rightarrow [_{c_i} \ active_i \ [_{c_i} \ fusion$$

The next rule can be fired only after the preceding ones and fuses the trash membrane with the copy of membrane c_i containing the $fusion$ object:

$$[_{trash} \ \epsilon \ [_{c_i} \ fusion \rightarrow \ [_{trash} \ fused$$

Finally, after the fusion the control state moves to l_2 by applying the following rule:

$$l' \ [_{cs} \ fused \rightarrow l_2 \ [_{cs} \ \epsilon$$

Notice that if we simulate a test for zero on counter i and the membrane c_i contains at least one object o then the *trash* membrane contains at least one object o after the simulation. Furthermore there is no rule that decreases the number of o in *trash*. Hence, *trash* remains empty while the PBFC system simulates correctly the counter machine. So, the state with control state l and counter c_i equal to v_i is reachable iff the configuration where the root node contains l, its child c_i contains v_i instances of o (and one object $active_i$) and an empty trash membrane is reachable. □

Theorem 1 shows the power of PBFC systems: they have the same expressive power as Turing machines when considering reachability of a particular configuration as accepting condition. However, as we prove in the remainder of the paper and contrary to Turing machines, some interesting properties of PBFC systems can be automatically checked. In particular, we concentrate on two other important decision problems, i.e., boundedness and coverability of a configuration. The boundedness problem poses a basic question on the behavior of a system, i.e., the finiteness of its state space.

Definition 2 (Boundedness). *Given a PBFC system Π with the initial configuration (M_0, μ_0), the* boundedness problem *consists in deciding if the set of configurations that are reachable from (M_0, μ_0) is finite.*

The coverability problems is a weaker form of reachability often used for qualitative analysis and verification of infinite-state systems [1, 9]. Instead of checking if a specific configuration is reachable or not, coverability is defined as the reachability of a (typically infinite) set of a configurations that have certain patterns. The most convenient way to formalize this idea is to introduce an ordering on configurations and formulate the coverability problem with respect to upward closed sets of configurations. Since PBFC configurations have a tree structure, it seems natural to consider here patterns of trees and, thus, to introduce an ordering on trees (with multisets of objects as labels). Specifically, we use here the tree embedding (a.k.a. Kruskal) order $\leq_K$ over trees [13].

Definition 3 (Tree Embedding). *Let M and M' be two trees with the set of nodes S and S', respectively; and assume a quasi order $\sqsubseteq$ on labels of tree nodes. Then, $M \leq_K M'$ iff there exists an injection $\rho : S \mapsto S'$ such that (i) for all $n \in S, n \sqsubseteq \rho(n)$ and (ii) for all $n, n' \in S$, we have that n' is in the sub-tree rooted by n iff $\rho(n')$ is in the sub-tree rooted by $\rho(n)$.*

In the case of PBFC configurations, the order $\sqsubseteq$ is defined as follows. Given a node n of a configuration (M, μ) and a node n' of (M', μ'), $n \sqsubseteq n'$ iff $\mathsf{type}(n) = \mathsf{type}(n')$ and $\mu(n) \preceq \mu'(n')$.

From the Kruskal tree theorem [13] (the version for unordered trees can be found in [5]), we know that if $\sqsubseteq$ is a well-quasi ordering (wqo) then $\leq_K$ is also a wqo (see preliminaries for def. of wqo). By Dickson's lemma [8], the order $\sqsubseteq$ is a wqo. Thus, the order $\leq_K$ is a wqo over PBFC configurations. The coverability problem is defined then as follows:

Definition 4 (Coverability). *Given a PBFC system Π with the initial configuration (M_0, μ_0) and a configuration (M, μ), the* $\leq_K$-coverability problem *consists in checking if there is a configuration (M', μ') which is reachable from (M_0, μ_0) and such that $(M, \mu) \leq_K (M', \mu')$.*

The intuition here is that the configuration (M, μ) defines the pattern of the set of configurations for which we ask the reachability question. A pattern is defined here as a tree with certain objects in each node.

A note about PBFC systems and (extended) Petri nets. In [6], it is shown that PB-systems can be encoded into (mimicked by) Petri nets. A Petri net is composed of a finite set of places P and a finite set of transitions T. A Petri net configuration $m : P \mapsto \mathbb{N}$, called marking, assigns $m(p)$ (black) tokens to each place $p \in P$. Each transition $t \in T$ removes/adds a fixed number of tokens from each place $p \in P$ (see [20] for a more detailed description of Petri nets). For instance, a transition may remove one token from a place p_1 and add one token into another place p_2. The boundedness problem and the $\preceq$-coverability problem are defined as in the case of PBFC systems where the order $\preceq$ over markings is the pointwise extension of the order $\leq$ over $\mathbb{N}$.

When adding fusion and clonation to PB systems, transfer Petri nets can be simulated. Those models are Petri nets extended with transfer arcs that move all the tokens contained in one place to another in one step. Hence, the number of tokens transferred is not fixed *a priori*. Since transfer Petri nets are more expressive than Petri nets [11], contrary to PB systems, PBFC systems cannot be encoded into (mimicked by) Petri nets.

A transfer Petri net $\mathcal{N}$ is encoded into a PBFC system as follows. For each place p we have a membrane name/type p. We also have a membrane name/type $\mathcal{N}$. A marking m is encoded into a configuration composed of a root membrane of name/type $\mathcal{N}$ which has, for each place p, two children of name/type p. The first child of type p contains only an object w_p and is used to simulate transfers from the place p. The second child contains one object ok_p and as many objects $\bullet$ as the number of tokens assigned by m to p, i.e., it is used to encode the content of the place p. The root membrane contains an object that describes the state of the simulation: the PBFC systems is ready to simulate a new transition or it is simulating a transition. Fig. 1 shows how a transfer from place p to p' is simulated: the membrane of type p encoding the content of the place p is fused with the membrane of type p' encoding the content of the place p'. Moreover, the other membrane of name/type p is cloned and the new copy is used to encode the content of the place p after the transfer.

Let r be the object contained into the root membrane of type $\mathcal{N}$ when the PBFC system is ready to simulate a new transition and assume that the transition t is only composed of a transfer from the place p to p'. Then, the transition t is simulated with the following rules.

The first two rules clone the membrane of type p containing the object w_p:

$$
\begin{array}{l}
r\ [_{\mathcal{N}}\ w_p \ \rightarrow\ r_1\ [_{\mathcal{N}}\ c_p \\
[_p\ c_p \ \rightarrow\ [_p\ w_p\ [_p\ cloned_p
\end{array}
$$

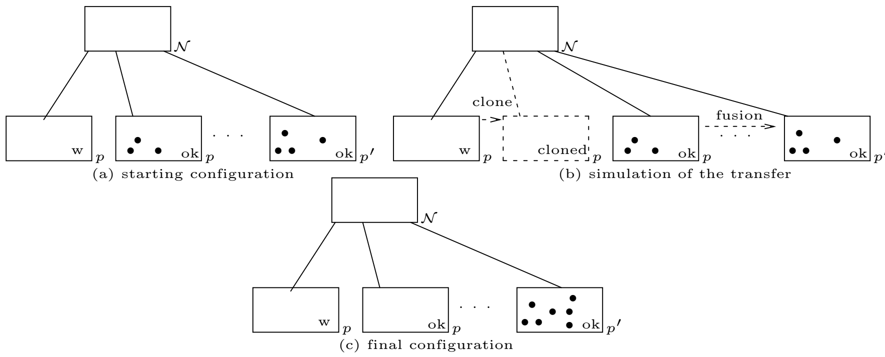

Fig. 1. Simulation of a transfer with a PBFC systems. Rectangles represents membranes.

The three next rules fuse the membranes encoding the content of the places p and p' into a unique membrane of type p' (encoding the content of the place p' after the transfer). Those rules can be applied only after the two previous ones.

$$\begin{array}{lcl} r_1 \; [_{\mathcal{N}} \; cloned_p & \rightarrow & r_2 \; [_{\mathcal{N}} \; cloned_p \\ r_2 \; [_{\mathcal{N}} \; ok_p & \rightarrow & r_3 \; [_{\mathcal{N}} \; f_{pp'} \\ [_p \; f_{pp'} \; [_{p'} \; ok_{p'} & \rightarrow & [_{p'} \; fused_{p'} \end{array}$$

Finally, the cloned membrane of name/type p becomes the membrane encoding the content of the place p and the result of the fusion encodes p'.

$$\begin{array}{lcl} r_3 \; [_{\mathcal{N}} \; fused_{p'} & \rightarrow & r_4 \; [_{\mathcal{N}} \; ok_{p'} \\ r_4 \; [_{\mathcal{N}} \; cloned_p & \rightarrow & r \; [_{\mathcal{N}} \; ok_p \end{array}$$

4 Decidability of Boundedness for extended PB systems

In this section, we prove that the boundedness problem is decidable for PBFC systems. To achieve that goal, we use the theory of well-structured transition systems [1, 9]. In order to apply results provided in [1, 9], we must first prove that PBFC systems together with a wqo over configurations form well-structured transition systems (see preliminaries for definitions).

We first notice that PBFC-systems with the Kruskal order are not well-structured. Indeed, consider only one type of membrane i and a boundary rule $r = a \; [_i \; b \rightarrow c \; [_i \; d$. Now consider two configurations (M, μ) and (M', μ') . The first one is composed of two nodes, the root and its unique child. The root contains the object a and its child contains b. Hence, r is applicable. The second configuration is similar to the first one except that there is an intermediate membrane between the root and its child. That intermediate membrane contains the object c, hence r is not applicable and the condition (ii) of the definition of wsts (monotonicity) does not hold.

Thus, we cannot directly use the theory of well-quasi ordering for PBFC and the order $\leq_K$ to solve the boundedness problem. Instead, we use another order, noted $\leq_D$, for which PBFC systems are strictly monotonic.

Assume two trees M and M' with S and S' as set of nodes, respectively, r and r' as root, respectively. Assume also a quasi order $\sqsubseteq$ on nodes of trees.

Definition 5 (The order $\leq_D$). *We say that $M \leq_D M'$ iff there exists an injection $\rho : S \mapsto S'$ such that $\rho(r) = r'$, for all $n \in S, n \sqsubseteq \rho(n')$ and for all $n, n' \in S$, we have that n' is a child of n iff $\rho(n')$ is a child of $\rho(n)$.*

In the case of PBFC configurations, the order $\sqsubseteq$ between labels of nodes is multiset inclusion as for $\leq_K$. For any pairs of configurations c and c', we use $c <_D c'$ to denote that $c \leq_D c'$ and $c' \not\leq_D c$.

Now, we say that a configuration (M, μ) is b-depth bounded if the depth (= max number of nested membranes) d of M is less of equal than b. Given a PBFC system (Γ, N, M, R, μ_0), let $b \in \mathbb{N}$ be the depth of M. Then, it is easy to check that all the reachable configurations are b-depth bounded. In the following, we say that the PBFC system is b-depth bounded and we restrict the set of configurations to b-depth bounded configurations. Based on this observation, we prove that $\leq_D$ is a wqo for the set of b-depth bounded configurations.

Lemma 1. *Given a (b-depth bounded) PBFC system Π, the quasi order $\leq_D$ over (b-depth bounded) configurations of Π is a wqo.*

Proof. To prove the lemma, we annotate each node n by its depth $\mathsf{Depth}(n)$. Then, $\leq_D$ corresponds to the Kruskal order where the order $\sqsubseteq$ on nodes is defined as follows: $n \sqsubseteq n'$ iff $\mathsf{Depth}(n) = \mathsf{Depth}(n')$, $\mathsf{type}(n) = \mathsf{type}(n')$ and $\mu(n) \preceq \mu(n')$. Since $\preceq$ is a wqo, the size of the set N of possible membrane names/types and the depth of configurations are bounded, we get that $\sqsubseteq$ is a wqo. Hence, applying the Kruskal theorem [13], we get that $\leq_D$ is a wqo. □

Thus, when reasoning on b-depth bounded configurations we can replace the tree embedding order $\leq_K$ with the tree immersion $\leq_D$. Second, we show that PBFC systems are strictly monotonic with respect to $\leq_D$.

Lemma 2. *PBFC systems are strictly monotonic with respect to $\leq_D$.*

Proof. Assume a b-depth bounded PBFC system and two (b-depth bounded) configurations $c = (M, \mu)$ and $c' = (M', \mu')$ such that $c \leq_D c'$. Assume also that the configuration $c'' = (M'', \mu'')$ is reachable from c by applying a rule. We show that, by applying the same rule on c' we get a configuration $c''' = (M''', \mu''')$ such that $c'' \leq_D c'''$. More precisely, from an injection ρ that shows that $c \leq_D c'$ we build an injection ρ' that shows that $c'' \leq_D c'''$. Furthermore, if $c <_D c'$ then $c'' <_D c'''$.

- *Internal rule.* First, notice that the tree structure of configurations does not change when applying an internal rule. Furthermore, if an internal rule $[_i\ u \rightarrow [_i\ v$ is applied on the node n, then the internal rule can also be applied on $\rho(n)$ since (a) $i = \mathsf{type}(n) = \mathsf{type}(\rho(n))$ and (b) $\mu(n) \preceq \mu'(\rho(n))$.

Moreover, from (b) we deduce that $(\mu(n) \ominus u) \oplus v \preceq (\mu'(\rho(n)) \ominus u) \oplus v$. Hence, by defining $\rho' = \rho$ we conclude that $c'' \leq_D c'''$. Finally, since the tree structure does not change we have that if M' has strictly more nodes than M then M''' has strictly more nodes than M''. Furthermore, if M contains a node n' such that $\mu(n') \prec \mu'(\rho(n'))$ then $\mu''(n') \prec \mu'''(\rho'(n'))$. Hence, we conclude that $c <_D c'$ implies that $c'' <_D c'''$;

- *Boundary rule.* First, notice that the tree structure does not change when applying a boundary rule. Furthermore, if a boundary rule $u\ [_i\ v \rightarrow u'\ [_i\ v'$ is applied on the node n and one of its child m, then the boundary rule can also be applied on $\rho(n)$ and $\rho(m)$ since (a) $i = \mathsf{type}(n) = \mathsf{type}(\rho(n))$, (b) $\mu(n) \preceq \mu'(\rho(n)), (c) \mu(m) \preceq \mu'(\rho(m))$ and (d) $\rho(m)$ is a child of $\rho(n)$. Moreover, from (b) and (c) we get that $(\mu(n) \ominus u) \oplus v \preceq (\mu'(\rho(n)) \ominus u) \oplus v$ and $(\mu(m) \ominus u) \oplus v \preceq (\mu'(\rho(m)) \ominus u) \oplus v$. Hence, by defining $\rho' = \rho$ we conclude that $c'' \leq_D c'''$. Since the tree structure of configurations does not change we have that if M' has strictly more nodes than M then M''' has strictly more nodes than M''. Furthermore, if M contains a node n' such that $\mu(n') \prec \mu'(\rho(n'))$ then $\mu''(n') \prec \mu'''(\rho'(n'))$. Hence, we conclude that $c <_D c'$ implies that $c'' <_D c'''$;
- *Fusion rule.* Assume that a fusion rule $[_i\ u\ \ [_j\ v\ \ \rightarrow [_k\ w$ is applied on nodes n with $\mathsf{type}(n) = i$ and n' with $\mathsf{type}(n') = j$. Then, the fusion rule can also be applied on $\rho(n)$ and $\rho(n')$ since (a) $i = \mathsf{type}(n) = \mathsf{type}(\rho(n))$, (b) $\mu(n) \preceq \mu'(\rho(n))$, (c) $j = \mathsf{type}(n') = \mathsf{type}(\rho(n'))$, (d) $\mu(n') \preceq \mu'(\rho(n'))$. Moreover, n and n' have the same father, so do $\rho(n)$ and $\rho(n')$ following the definition of $\leq_D$. From (b) and (c) we get that $(\mu(n) \ominus u) \oplus (\mu(n') \ominus v) \oplus w \preceq (\mu'(\rho(n)) \ominus u) \oplus (\mu'(\rho(n')) \ominus v) \oplus v$. Assume that fusion creates the new node n'' in c'' and n''' in c'''. We define the injection ρ' as follows: $\rho'(m) = \rho(m)$ for nodes $m \neq n''$, and $\rho'(n'') = n'''$. Notice that the number of nodes always decreases by 1 when applying a fusion rule. Hence, if the number of nodes in M is strictly smaller than in M' then the number of nodes in M'' is strictly smaller than in M'''. Furthermore, if c contains a node $m \notin \{n, n'\}$ such that $\mu(m) \prec \mu'(\rho(m))$ then $\mu''(m) \prec \mu'''(\rho'(m))$ and if $\mu(n) \prec \mu'(\rho(n))$ or $\mu(n') \prec \mu'(\rho(n'))$ then $\mu''(n'') \prec \mu'''(n''')$. Hence, we conclude that $c <_D c'$ implies that $c'' <_D c'''$;
- *Clonation rule.* Assume that a clonation rule $[_i\ u \rightarrow [_i\ v\ \ [_i\ w$ is applied on a node n. Then, the clonation rule can also be applied on $\rho(n)$ since (a) $i = \mathsf{type}(n) = \mathsf{type}(\rho(n))$ and (b) $\mu(n) \preceq \mu'(\rho(n))$. Furthermore, from (b) we get that $(\mu(n) \ominus u) \oplus v \preceq (\mu'(\rho(n)) \ominus u) \oplus v$ and $(\mu(n) \ominus u) \oplus w \preceq (\mu'(\rho(n)) \ominus u) \oplus w$. Hence, we define the mapping ρ' as follows: $\rho'(m) = \rho(m)$ for all the nodes m appearing in M. For each new node m which is a copy of the node m' in M'', we defined $\rho'(m) = m''$ where m'' is the copy of the node m''' in M''' such that $\rho(m') = m'''$. Notice that the number of nodes added in M to obtain M'' is always smaller than or equal to the number of nodes added in M' to obtain M'''. Hence, if the number of nodes in M is strictly smaller than in M' then the number of nodes in M'' is strictly smaller than in M'''. Furthermore, if M contains a node n such that $\mu(n) \prec \mu'(\rho(n))$ then $\mu''(n) \prec \mu'''(\rho'(n))$. Hence, we conclude that $c <_D c'$ implies that $c'' <_D c'''$. $\square$

From Lemma 1 and 2, we conclude that PBFC systems are strictly monotonic wsts.

Proposition 1. *PBFC systems with the order $\leq_D$ are strictly monotonic wsts.*

The algorithm of [9] that solves the boundedness problem for wsts computes a finite prefix of the reachability tree. The unfolding of a branch is stopped as soon as from the current configuration c either there is no configuration c' with $c \to c'$, i.e., c is a deadlock, or there is another configuration c' in the same branch such that $c' \leq_D c$. The set of reachable configurations is infinite if and only if there is a leaf c with a predecessor c' such that $c' <_D c$. Since we can compute the successors for any configuration and the order $\leq_D$ is decidable, we can instantiate that algorithm to PBFC systems.

Using the construction given in Section 3, we can reduce the boundedness problem for transfer Petri nets to the boundedness problem for PBFC systems (in polynomial time). By adapting the construction presented in [19], we can also show that the boundedness problem for transfer Petri nets has nonprimitive recursive complexity. Hence we have the following theorem.

Theorem 2. *The boundedness problem is decidable for PBFC systems and has nonprimitive recursive complexity.*

5 Decidability of Coverability for extended PB systems

We now apply the theory of well-structured transition systems [1, 9] to solve the $\leq_K$-coverability problem for PBFC systems. Since PBFC systems together with wqo $\leq_K$ do not form a wsts, we cannot apply directly the results of [1, 9]. Instead, we reuse the wqo $\leq_D$ introduced into the previous section. The $\leq_D$*-coverability problem* is defined similarly to the $\leq_K$-coverability problem by replacing $\leq_K$ by $\leq_D$. In the remainder of the section, we show how to instantiate the algorithm proposed in [1, 9] to solve the $\leq_D$-coverability problem. Then, we show how to use that result to solve the $\leq_K$-coverability problem.

5.1 Solving the $\leq_D$-coverability problem

The algorithm in [1, 9] that solves the coverability problem manipulates *upward-closed sets* of configurations. Remember that we restrict the set of configurations of a b-depth bounded PBFC system to b-depth bounded configurations. A set of PBFC configurations U is upward-closed if and only if for any pair of configurations c, c', we have that $c \in U$ and $c \leq_D c'$ implies that $c' \in U$. Since $\leq_D$ is a wqo (and a partial order) over b-depth bounded configurations, any upward-closed set U of b-depth bounded configurations has a finite set of minimal elements, i.e., one finite and minimal (with respect to $\subseteq$) set B such that $U = \{c \mid \exists c' \in B : c' \leq_D c\}$. Given a set of b-depth bounded configurations F, $F\uparrow$ denotes the upward-closed set of b-depth bounded configurations $\{c \mid \exists c' \in F : c' \leq_D c\}$. If F is finite then we say that it is a *finite basis* of $F\uparrow$.

Notice that a finite basis may contain useless elements, ie. configurations c such that there exists another configuration c' with $c' <_D c$.

More precisely, the algorithm of [1, 9] manipulates upward-closed sets towards one of their finite basis. To instantiate that algorithm we need thus to show how to compute the operations needed by that algorithm on that symbolic representation. Given an upward-closed set U, let $\mathsf{Pre(U)} = \{\mathsf{c} \mid \mathsf{c} \to \mathsf{c'}, \mathsf{c'} \in \mathsf{U}\}$. Roughly speaking, the algorithm computes the sequence of upward-closed sets $(\mathcal{P}_i)_{i\geq 0}$ such that $\mathcal{P}_0 = U$ and $\forall i \geq 1, \mathcal{P}_i = \mathcal{P}_{i-1} \cup \mathsf{Pre}(\mathcal{P}_{\mathsf{i-1}})$ until it converges. Hence, we need an algorithm that computes, for any upward-closed set U, a finite basis of the set $\mathsf{Pre(U)}$. Before exhibiting such an algorithm, we show that a finite basis of the intersection of two upward-closed sets is computable [2].

Lemma 3. *There exists an algorithm that builds, given two b-depth bounded configurations c and c′ of a b-depth bounded PBFC system Π, a finite basis of the upward-closed set of b-depth bounded configurations* $\{c\}\uparrow \cap \{c'\}\uparrow$.

Proof. Given two b-depth bounded configurations $c = (M, \mu)$ and $c' = (M', \mu')$, we show here how to build a finite set of configurations $\mathcal{M}$ such that $\mathcal{M}\uparrow = \{(M, \mu)\}\uparrow \cap \{(M', \mu')\}\uparrow$.

Let S and S' be the set of nodes of M and M', respectively, and r and r' be the root node of M and M', respectively. We define Π as the set of partial injection $\rho : S \mapsto S'$ such that (i) $\rho(r) = r'$ and (ii) for all $n, n' \in S$, n is the parent of n' and $\rho(n')$ is defined implies that $\rho(n)$ is defined and $\rho(n)$ is the parent of $\rho(n')$. Notice that the set Π is finite and computable.

Then, for all $\rho \in \mathcal{M}$, we defined the b-depth bounded configuration (M_ρ, μ_ρ) as follows. The set of nodes of the tree M_ρ is

$$\begin{aligned} S_\rho = &\{\{n\} \mid n \in S, \rho(n) \text{ is undefined }\}\cup \\ &\{\{n\} \mid n \in S', \rho^{-1}(n) \text{ is undefined }\}\cup \\ &\{\{n, n'\} \mid n \in S, n' \in S', \rho(n) = n'\} \end{aligned}$$

where ρ^{-1} is the inverse of ρ. A node $n' \in S_\rho$ is a child of a node $n \in S_\rho$ iff there exist $m \in n, m' \in n'$ and m' is a child of m either in M or in M'. Finally, μ_ρ is defined as follows. For all $m \in S_\rho$, one of the following holds:

- $m = \{n\}$ with $n \in S$ and $\mu_\rho(m) = \mu(n)$;
- $m = \{n\}$ with $n \in S'$ and $\mu_\rho(m) = \mu'(n)$;
- $m = \{n, n'\}$ with $n \in S, n' \in S'$ and for all $\gamma \in \Gamma$ we have that $\mu_\rho(m)(\gamma) = max(\mu(n)(\gamma), \mu'(n')(\gamma))$.

Again, notice that for any $\rho \in \mathcal{M}$ the b-depth bounded configuration (M_ρ, μ_ρ) is computable. Finally, notice that we have that $\cup_{\rho\in\mathcal{M}}\{(M_\rho, \mu_\rho)\}\uparrow = \{(M, \mu)\}\uparrow \cap\{(M', \mu')\}\uparrow$. □

Proposition 2. *There exists an algorithm that builds, given a b-depth bounded PBFC system Π and a finite basis of an upward closed set U of b-depth bounded configurations, a finite basis of the set* $\mathsf{Pre(U)}$.

[2] Notice that the intersection of upward-closed sets is always an upward-closed set.

Proof. Let F be a finite basis of U and F' be the finite basis of $\mathsf{Pre}(\mathsf{U})$ computed by the algorithm. The algorithm is defined by cases as follows.

Internal rule. Assume an internal rule $[_i\ u \to [_i\ v$ and $c = (M, \mu) \in F$. We consider two cases. Either the rule is backwardly applied on a node which is explicitly in c or not. To take into account the first case, for any node n with $\mathsf{type}(n) = i$ [3] we have that F' contains the configuration (M, μ') such that $\mu'(n) = (\mu(n) \ominus v) \oplus u$, and $\mu'(n') = \mu(n')$ for all $n' \neq n$. For the second case, we must ensure that the predecessors have a node containing the objects u. Notice that the new node may appear an any depth (bounded by b). Hence, F' contains any b-depth bounded configurations (M', μ') obtained from c by adding a sub-tree composed of nodes $n_1, \ldots, n_k$ $(k \geq 1)$ such that for all $1 \leq j < k$, $\mathsf{type}(n_j)$ is any membrane type, $\mu'(n_j) = \epsilon$ and n_j has n_{j+1} as unique child; $\mathsf{type}(n_k) = i$ and $\mu'(n_k) = u$.

Boundary rule. Assume a boundary rule $u\ [_i\ v \to u'\ [_i\ v'$ and $c = (M, \mu) \in F$. We consider three cases:

- The rule is applied on two nodes that appear explicitly in c. In that case, for any pairs of nodes n, n' of c such that $\mathsf{type}(n) = i$ and n' is a child of n the following holds. We have a predecessor configuration that contains a node m with $\mathsf{type}(m) = i$ with one child m' such that m contains the objects $(\mu(n) \ominus u') \oplus u$ and m' contains the objects $(\mu(n') \ominus v') \oplus v$. More precisely, F' contains the configuration (M, μ') where $\mu'(n) = (\mu(n) \ominus u') \oplus u$, $\mu'(n') = (\mu(n') \ominus v') \oplus v$, and $\mu'(n'') = \mu(n'')$ for all $n'' \notin \{n, n'\}$;
- The rule is applied on one node of type i that explicitly appears in c and on one of its child that does not appear explicitly. In that case, for any node n of c such that $\mathsf{type}(n) = i$, the following holds. We have predecessor configurations that contain a node m with $\mathsf{type}(m) = i$ containing the objects $(\mu(n) \ominus u') \oplus u$, and which has one child containing the objects v. More precisely, F' contains a configuration (M', μ') where M' is obtained from M by adding one new child n' to n where $\mathsf{type}(n')$ is any membrane type; and μ' is such that $\mu'(n) = (\mu(n) \ominus u') \oplus u$, $\mu'(n') = v$ and $\mu'(n'') = \mu(n'')$ for all $n'' \notin \{n, n'\}$;
- Finally, the rule may be applied on two nodes that do not explicitly appear in c. In that case, the predecessor configurations must contain two nodes containing the objects u and v, respectively. As in the case of internal rules, those nodes may appear at any depth (smaller than or equal to b). Hence, F' contains any b-depth bounded configurations (M', μ') obtained from c by adding nodes $n_1, \ldots, n_k$ $(k \geq 2)$ such that for all $1 \leq j < k-1$, $\mathsf{type}(n_j)$ is any type, $\mu'(n_j) = \epsilon$ and n_j has n_{j+1} as unique child; $\mathsf{type}(n_{k-1}) = i$, $\mu'(n_{k-1}) = u$ and n_{k-1} has n_k as unique child; $\mathsf{type}(n_k)$ is any type and $\mu'(n_k) = v$.

Fusion rule. Assume a fusion rule $[_i\ u\ \ [_j\ v\ \to [_k\ w$. We consider two cases:

[3] We do not impose that $v \preceq \mu(n)$ because some elements of v may not appear explicitly in $\mu(n)$.

- The fused sub-tree does not appear explicitly into c. In that case, we must ensure into the predecessor configurations that at least one node n has one child n' with $\mathsf{type}(n') = i$ containing the objects u and one child n'' with $\mathsf{type}(n'') = j$ containing the objects v. Notice that those nodes may appear at any depth smaller than or equal to b. More precisely, F' contains all the b-depth bounded configurations (M', μ') where M' is built from M by adding nodes $n_1, \ldots, n_k, n_{k+1}$ $(k \geq 1)$ such that n_1 is a child of a node n_0 appearing into M, for all $1 \leq i \leq k-1$, n_i is a child of n_{i-1} and $\mathsf{type}(n_i)$ is any membrane type; and n_k and n_{k+1} are children of n_{k-1}, $\mathsf{type}(n') = i$ and $\mathsf{type}(n'') = j$. Furthermore, $\mu'(n_k) = u, \mu'(n_{k+1}) = v$, for all $1 \leq i \leq k-1, \mu'(n_i) = \epsilon$, and for all node $n'' \notin \{n_1, \ldots n_{k+1}\}$ we have $\mu'(n'') = \mu(n'')$;
- The fused sub-tree appears explicitly into c. In that case, M contains a node n with $\mathsf{type}(n) = k$. Then, F' contains all the configurations (M', μ') such that M' is built from M by replacing the sub-tree rooted by n by two sub-trees rooted by the nodes n' with $\mathsf{type}(n') = i$ and n'' with $\mathsf{type}(n'') = j$. The sub-trees rooted by children of n' and n'' form the set of sub-trees rooted by children of n. Furthermore, the function μ' satisfies the following: $\mu'(n')$ and $\mu'(n'')$ are multi-sets of objects such that $u \preceq \mu'(n'), v \preceq \mu'(n'')$ and $(\mu'(n') \ominus u) \oplus (\mu'(n'') \ominus v) = \mu(n) \ominus w$, and for all $m \notin \{n', n''\}$, $\mu'(m) = \mu(m)$.

Clonation rule. Assume a rule $[_i\ u \quad \to [_i\ v \quad [_i\ w$. We consider several cases:

- The cloned sub-tree and the resulting sub-tree of the clonation do not appear into c. In that case, the predecessor configurations must contain a node of type i containing the objects u at any depth smaller than or equal to b. More precisely, F' contains any configuration (M', μ') obtained from (M, μ) by adding nodes $n_1, \ldots, n_k (k \geq 1)$ such that for all $1 \leq j < k, \mathsf{type}(n_j)$ is any type, $\mu'(n_j) = \epsilon$ and n_j has n_{j+1} as unique successor; $\mathsf{type}(n_k) = i$ and $\mu'(n_k) = u$;
- The cloned sub-tree appears explicitly into c but the resulting sub-tree of the clonation does not. For any node n of c such that $\mathsf{type}(n) = i$ F' contains a configuration (M, μ') where $\mu'(n) = (\mu(n) \ominus v) \oplus u$ and $\mu'(n') = \mu(n')$ for all $n' \neq n$;
- The resulting sub-tree of clonation appears explicitly into c but the cloned sub-tree does not. For any node n of c such that $\mathsf{type}(n) = i$ F' contains a configuration (M, μ') where $\mu'(n) = (\mu(n) \ominus w) \oplus u$ and $\mu'(n') = \mu(n')$ for all $n' \neq n$;
- The cloned sub-tree and the resulting sub-tree of clonation appear explicitly into c. For any nodes n and n' such that n and n' have the same parent n'', $\mathsf{type}(n) = i$ and $\mathsf{type}(n') = i$, we have the following. Let the configuration (M, μ') such that $\mu'(n) = (\mu(n) \ominus v), \mu'(n') = (\mu(n') \ominus w)$, and $\mu'(m) = \mu(m)$ for all $m \neq n, n'$. Assume that (M_n, μ_n) and $(M_{n'}, \mu_{n'})$ are the configurations corresponding to the sub-trees rooted by n and n', respectively, and let $\mathcal{B}$ be a finite basis of the set $\{(M_n, \mu_n)\}\uparrow \cap \{(M_{n'}, \mu_{n'})\}\uparrow$. Notice that the set $\mathcal{B}$ is computable following Lemma 3. For any $(M_\cap, \mu_\cap) \in \mathcal{B}$, F' contains a configuration (M''', μ''') built from (M', μ') by removing the sub-trees rooted by n and n' and by adding the tree $M_\cap$ as a sub-tree of n''. The mapping

μ''' is similar to μ' for the nodes that does not appear in $M_\cap$, for the root r of $M_\cap$ we have that $\mu'''(r) = \mu_\cap(r) \oplus u$, and for the other nodes $m \neq r$ of $M_\cap$ we have that $\mu'''(m) = \mu_\cap(m)$. □

Notice that the union of upward-closed sets is computed by making the union of their finite basis, and the convergence of the sequence $(\mathcal{P}_i)_{i\geq 0}$ can be decided by testing $\leq_D$ on elements of finite basis. Since $\leq_D$ is decidable, from Lemma 2 and Proposition 2 we get the decidability of the $\leq_D$-coverability problem.

Again, the encoding given in Section 3 allows us to reduce the $\preceq$-coverability problem for transfer Petri nets to the $\leq_D$-coverability problem for PBFC systems (in polynomial time) and a trivial modification of the construction given in [19] allows us to prove that the coverability problem for transfer Petri nets has a nonprimitive recursive complexity. Hence, the next proposition.

Proposition 3. *The $\leq_D$-coverability problem is decidable for PBFC systems and has a nonprimitive complexity.*

5.2 Decidability of the $\leq_K$-coverability problem

We now show how to reduce the $\leq_K$-coverability problem to the $\leq_D$-coverability problem. Let a b-bounded depth configuration $c = (M, \mu)$ and $\mathcal{M}(c, b)$ be the set of b-depth bounded configurations (M', μ') built from c by adding possibly intermediate nodes between nodes of M. Those intermediate nodes n have any membrane type and are such that $\mu'(n) = \epsilon$ and have exactly one successor, either another intermediate node or a node appearing in M. In other words, the configurations in $\mathcal{M}(c, b)$ have a similar structure than c except that the nodes of M are potentially at a larger depth (bounded by b).

By definition, any b-depth bounded configuration (M'', μ'') has a depth bounded by $b \in \mathbb{N}$. Hence, we have that $(M, \mu) \leq_K (M'', \mu'')$ if and only if there exists a b-depth bounded configuration $(M', \mu') \in \mathcal{M}(c, b)$ such that $(M', \mu') \leq_D (M'', \mu'')$. Since the set $\mathcal{M}(c, b)$ is finite and computable, we can solve the $\leq_K$-coverability problem for a b-depth bounded PBFC system Π and a b-depth bounded configuration (M, μ) by solving $\leq_D$-coverability problems for Π and configurations in $\mathcal{M}(c, b)$.

Finally, the encoding of transfer Petri nets into PBFC systems given in Section 3 and [19] allow us to conclude that the complexity of the $\leq_K$-coverability problem is nonprimitive recursive.

Theorem 3. *The $\leq_K$-coverability problem is decidable for PBFC systems and has a nonprimitive recursive complexity.*

6 Conclusion

In this paper we have investigated the computational property of a biologically inspired model providing the notion of membrane and rules for fusion of clonation of membranes. The resulting model, called PBFC, is an extension of

PB-systems [3]. PBFC systems seem to be an interesting example of an unconventional computing model in between Turing machines and Petri nets. Indeed, they provide non-trivial built-in operations for the manipulation of tree structured collections of objects (reachability is undecidable). However, as for Petri nets, decision problems that are strictly related to the verification of structural and qualitative properties like boundedness and coverability turn out to be decidable for PBFC systems.

References

1. P. A. Abdulla, K. Čerāns, B. Jonsson, and T. Yih-Kuen. General decidability theorems for infinite-state systems. LICS '96: 313–321.
2. A. Alhazov, R. Freund, and A. Riscos-Núñez. Membrane division, restricted membrane creation and object complexity in P systems. Computer Mathematics 83(7): 529 - 547, 2006.
3. F. Bernardini and V. Manca P systems with boundary rules. WMC 2002: 107-118.
4. D. Besozzi, C. Zandron, G. Mauri and N. Sabadini. P systems with gemmation of mobile membranes. ICTCS 2001: 136-153.
5. M. Bezem, J. W. Klop, and R. de Vrijer. Term rewriting systems. Cambridge University Press, 2003
6. S. Dal Zilio and E. Formenti. On the dynamics of PB systems: A Petri net view. WMC 2003: 153-167.
7. G. Delzanno and L. Van Begin. On the dynamics of PB systems with volatile membranes. WMC 2007: 240-256.
8. L. E. Dickson. Finiteness of the odd perfect and primitive abundant numbers with distinct factors. Amer. J. Math. 35: 413-422, 1913.
9. A. Finkel and Ph. Schnoebelen. Well-structured transition systems everywhere! TCS 256(1-2): 63-92, 2001.
10. G. Franco and V. Manca. A membrane system for the leukocyte selective recruitment. WMC 2003: 181-190.
11. G. Geeraerts, J.-F. Raskin and L. Van Begin. Well-structured languages. Acta Informatica 44(3-4): 249-288, 2007.
12. O. H. Ibarra, Z. Dang, and Ö. Egecioglu. Catalytic P systems, semilinear sets, and vector addition systems. TCS 312(2-3): 379-399, 2004.
13. J.B. Kruskal Well-quasi Ordering, The Tree Theorem, and Vazsonyi's conjecture. Trans. American Math. Soc. 95: 210-225, 1960.
14. C. Li, Z. Dang, O. H. Ibarra, H.-C. Yen. Signaling P systems and verification problems. ICALP 2005: 1462-1473
15. Gh. Pãun. Computing with membranes. J. of Computer and System Science 61(1): 108-143, 2000.
16. Gh. Pãun. P systems with active membranes: attacking NP-complete problems. J. of Automata, Languages, Combinatorics 6(1): 75-90, 2001.
17. Gh. Pãun, Y. Suzuki, H. Tanaka, and T. Yokomori. On the power of membrane division in P systems. TCS 324(1): 61 - 85, 2004.
18. A. Regev, E.M. Panina, W. Silverman, L. Cardelli and E. Shapiro. BioAmbients: an abstraction for biological compartments. TCS 325(1): 141-167, 2004.
19. Ph. Schnoebelen. Verifying lossy channel systems has nonprimitive recursive complexity. IPL 83(5): 251-261, 2002.
20. W. Reisig. Petri Nets - An Introduction. Springer, 1985.